Overview for Inverted Pendulum Demo

Tao Luo, taoluo@engineering.uiuc.edu
Jaehoon Ha, jha4@engineering.uiuc.edu
Spencer Hoke, hoke@uiuc.edu
Marco Caccamo, mcaccamo@uiuc.edu

January 27, 2005

Contents

1 INTRODUCTION **4**

2 GOALS **5**
2.1 MAIN GOAL . . . 5
2.2 SUBGOALS . . . 5
2.2.1 Stage 1 - DT2811 Control (IPC_DT2811) . . . 5
2.2.2 Stage 2 - Camera Control (IPC_Camera) . . . 5
2.2.3 Stage 3 - Remote Network Control (IPC_RC) . . . 5
2.2.4 Stage 4 - Network Camera Control (IPC_NETCAM) . . . 5
2.2.5 Stage 5 - FTIEDF Mote Control (IPC_MOTECAM) . . . 5

3 EQUIPMENT **6**
3.1 Total Available . . . 6
3.2 Stage 1 - DT2811 Control (IPC_DT2811) . . . 6
3.3 Stage 2 - Camera Control (IPC_Camera) . . . 6
3.4 Stage 3 - Remote Network Control (IPC_RC) . . . 6
3.5 Stage 4 - Network Camera Control (IPC_NETCAM) . . . 7
3.6 Stage 5 - FTIEDF Mote Control (IPC_MOTECAM) . . . 7

4 EQUIPMENT NOTES **8**
4.1 Computers . . . 8
4.2 DT2811 A/D Card . . . 8
4.3 Power Module PA0103 . . . 8
4.4 Track, Cart, Rod, and LED . . . 8
4.5 Framegrabber Cards . . . 9
4.6 Ethernet Cards . . . 10
4.7 Analog Cameras . . . 10
4.8 Motes . . . 10

5 S.Ha.R.K. DEVELOPMENT NOTES **12**
5.1 Important S.Ha.R.K. Types . . . 12
5.1.1 Main Task . . . 12
5.1.2 Hard Tasks . . . 12
5.1.3 Soft Tasks . . . 12
5.1.4 Cabs . . . 12
5.2 Adding Drivers (libdep.mk) . . . 13
5.3 DT2811 A/D Card Driver (dt2811.c, dt2811.h) . . . 13
5.4 Stage 1: IPC_DT2811 . . . 14
5.4.1 void waitenter . . . 14
5.4.2 int da_motor . . . 14
5.4.3 void calibrate . . . 14
5.4.4 float v2theta, v2x . . . 14
5.4.5 TASK readingReg . . . 14

5.4.6 TASK cart or cart_control . . . 14
5.5 Framegrabber Code (camera.c, camera.h) . . . 15
5.6 Stage 2: IPC_Camera . . . 15
5.6.1 TASK display_inputs . . . 16
5.7 Stage 3: IPC_RC . . . 16
5.8 Ethernet Network (network.c, network.h) . . . 16
5.8.1 FRAME *make_packet . . . 16
5.8.2 TASK network_send . . . 16
5.8.3 TASK network_receive . . . 16
5.9 Stage 4: IPC_NETCAM . . . 16
5.10 IPC_MOTECAM . . . 17
5.11 MOTES (motecom.c, motecom.h) . . . 17

6 TinyOS DEVELOPMENT NOTES 18
6.1 TinyOS . . . 18
6.1.1 Where to Get Help . . . 18
6.1.2 Using the Motes . . . 18
6.2 FAI-EDF Implementation . . . 18
6.2.1 Graphical Overview . . . 18
6.2.2 Schedule and Message Formats . . . 19
6.2.3 TOS_Msg Preparation . . . 19
6.2.4 Changing Messages . . . 19
6.2.5 Maximum Sizes . . . 19
6.2.6 LED Meanings . . . 20
6.2.7 Packet Format . . . 20
6.2.8 Changing Settings . . . 20
6.2.9 Changing Preamble Size . . . 20
6.2.10 When setting MAX_TX_TIME . . . 20
6.2.11 Real-Time Updating of Data from Serial Port . . . 21
6.2.12 Not Implemented in this Version . . . 21

7 RUNNING THE PROGRAMS 22
7.1 Stage 1: DT2811 . . . 22
7.2 Stage 2: Camera . . . 22
7.3 Stage 3: Remote Control . . . 22
7.4 Stage 4: Network Camera . . . 22
7.5 Stage 5: Motes . . . 22

8 TROUBLESHOOTING 23
8.1 Problems and Solutions . . . 23
8.1.1 Floppy disk died . . . 23
8.1.2 Rod is sticking when it's close to perpendicular . . . 23
8.1.3 Rod voltages changes after moving back to the same position . . . 23
8.1.4 Rod potentiometer range is not as desired . . . 23
8.1.5 Cart potentiometer range is not as desired . . . 23

8.1.6 Cart moves even when the program is not running 24
8.1.7 Cart balances, but then slowly moves off in one direction 24
8.1.8 Cart and rod stop balancing with working program 24
8.1.9 Ethernet network does not work 24
8.1.10 After exiting, the screen becomes gibberish 24
8.1.11 Restarting S.Ha.R.K freezes computer during S.Ha.R.K. initialization 25
8.1.12 The image from the framegrabber is not desired 25
8.1.13 Program freezes after a few seconds. May or may not quit 25
8.1.14 Program freezes or quits with a non-scheduling error 25
8.1.15 Mote appears to be on, but the computer isn't receiving anything . . 25
8.1.16 Program a mote gives a long list of flash errors 26

1 INTRODUCTION

This project originated from the need for a running demo using the Fault-Tolerant Implicit EDF scheduling algorithm for wireless communication. Since inverted pendulum systems are the standard for testing control algorithms, we felt it would be an excellent demonstration for the algorithm. Although S.Ha.R.K. does contain an official demo for controlling the inverted pendulum, it did not contain the functionality we sought. Therefore, we created our own demo in which we use camera sensors in complement with potentiometer sensors to balance the inverted pendulum. In this document are the tips and bits of wisdom we offer to students unfamiliar with our demo and/or inverted pendulums.

2 GOALS

2.1 MAIN GOAL

To stabilize the inverted pendulum using camera sensors transmitting data through wireless motes using the Fault-Tolerant Implicit EDF scheduling algorithm.

2.2 SUBGOALS

2.2.1 Stage 1 - DT2811 Control (IPC_DT2811)

After testing and configuring the base equipment, stabilize the inverted pendulum using two potentiometer sensors, one for angle of the pendulum and one for cart's position.

2.2.2 Stage 2 - Camera Control (IPC_Camera)

Adding the camera equipment, stabilize the inverted pendulum using one potentiometer sensor for the rod's angle and a camera for the cart's horizontal movement.

2.2.3 Stage 3 - Remote Network Control (IPC_RC)

Using the UDP network protocol, stabilize the inverted pendulum using the two potentiometer sensors read by a local I/O program and sent to a remote networked control program.

2.2.4 Stage 4 - Network Camera Control (IPC_NETCAM)

Using the UDP network protocol, stabilize the inverted pendulum using two switchable remote networked cameras and a local control program.

2.2.5 Stage 5 - FTIEDF Mote Control (IPC_MOTECAM)

Using wireless mote support, stabilize the inverted pendulum using two switchable remote networked cameras and a local control program.

3 EQUIPMENT

3.1 Total Available

S.Ha.R.K. 1.3, 1.4, 1.5 Alpha (available from http://shark.sssup.it/)
3 - Computers (2 with ISA slots for DT2811 card)
Dell - 350 MHz (ISA), Gateway - 200 MHz (ISA), C93847 - 1 GHz
2 - Framegrabber cards (Osprey 101 and Winfast TV2000)
2 - Analog cameras (Osprey 101 Webcam and Day/Night Security Camera)
1 - DT2811 A/D card and cables
3 - Ethernet cards
1 - Track (Painted Black)
1 - Cart (Painted Black)
1 - Rod (Black Taped)
1 - Power Module PA0103 (preconfigured)
1 - 2600 MCD Blue LED (for camera tracking)
TinyOS version 1.1.0 (available from http://www.tinyos.net/)
3 - MICA2 Motes
3 - Mote Programming Boards (for serial port connection) 1 - Manual Binder (AD1200 and S.Ha.R.K. 1.3 Manuals)
1 - Toolbox with level, screwdriver, black tape, etc.

3.2 Stage 1 - DT2811 Control (IPC_DT2811)

1 - Computer running S.Ha.R.K. 1.5 Alpha
1 - DT2811 A/D card and cables
Track & Cart & Rod & Power Module

3.3 Stage 2 - Camera Control (IPC_Camera)

2 - Computers running S.Ha.R.K. 1.5 Alpha
1 - Framegrabber card
1 - Analog camera
1 - DT2811 A/D card and cables
1 - LED
Track & Cart & Rod & Power Module

3.4 Stage 3 - Remote Network Control (IPC_RC)

2 - Computers running S.Ha.R.K. 1.5 Alpha
2 - Ethernet cards
1 - DT2811 A/D card and cables
Track & Cart & Rod & Power Module

3.5 Stage 4 - Network Camera Control (IPC_NETCAM)

3 - Computers running S.Ha.R.K. 1.5 Alpha
3 - Ethernet cards
2 - Framegrabber cards
2 - Analog cameras
1 - DT2811 A/D card and cables
1 - LED
Track & Cart & Rod & Power Module

3.6 Stage 5 - FTIEDF Mote Control (IPC_MOTECAM)

3 - Computers running S.Ha.R.K. 1.5 Alpha
3 - MICA2 Motes running TinyOS 1.1.0
3 - Mote programming boards 2 - Framegrabber cards
2 - Analog cameras
1 - DT2811 A/D card and cables
1 - LED
Track & Cart & Rod & Power Module

4 EQUIPMENT NOTES

This section contains notes about the equipment we used. This section will probably only be useful to you if you have the same equipment. However, it might be helpful to skim it at least because some of it might apply to you.

4.1 Computers

S.Ha.R.K. is quite picky about computers. Programs that run fine on one may not run at all on another. Other times, programs that execute fine on one computer may have problems on another. It is recommend to test your program frequently, and on multiple computers.

NOTE: Since S.Ha.R.K. is a real-time operating system, tasks that cannot be scheduled on one computer may be able on a faster one. The same applies in reverse. Keep that in mind when setting deadlines for tasks.

4.2 DT2811 A/D Card

This card uses an ISA slot, so whatever program you develop using this card has to run on a slow (PII or lower) system. The cable had already been wired to use the ground channel in the card to reduce the noise, so no further modifications were necessary. We labeled the colors of the wires corresponding to the card's pin outs in the AD1200 Manual.

4.3 Power Module PA0103

For us, this had been wired already, so we did not have to change anything.

4.4 Track, Cart, Rod, and LED

These were all wired and configured to work with the DT2811 card already, so no large modifications were necessary. An LED was taped onto the cart to provide an easy way for the cameras to track the cart. It was wired to the Power Module in series with resistors.

The track was set to be level with the ground. However, over time, it needs readjusting. It is recommended that you check before you start. In the toolbox, there should be a level for this job. Each foot of the base can be twisted to adjust the height, and the level of the track itself can be adjusted with the two screws (size 7/64" hex). We found the track dips in the center, so we stuck some paper underneath it. We painted the wood matte black to cancel the reflections from the bright LED. The track should be secured either by tape or other means to the table, since the movement of the cart will cause it to move also.

Two potentiometers (adjustable resistors) are built into the cart. The rod angle potentiometer is the big yellow cylinder on top of the cart. It can be adjusted by the screw in front of the cart. The second potentiometer can be found behind the big wheel of the cart. It can be adjusted by the screw holding the big wheel in place. As the rod and the wheel

moves, it changes the resistance in these potentiometers, which in turn changes the voltage returned to the DT2811 card. Once again, the DT2811 card can only take from -5 to 5 volts, so care must be taken to never exceed these ranges when running your inverted pendulum stabilizing program. To zero the cart, we recommend lifting the cart and turning the big wheel until it registers at 0 volts (S.Ha.R.K. program should show this). Then place it down on the center of the track. For the rod, place the level against the rod and adjust so the air bubble is in between the two lines.

If the rod voltage changes after moving it back to the original position, the rod receptacle might be too loose. There is a small star screw that can be tightened using a 3/32" Allen wrench. However, if over-tightened, the rod will not move freely and will stick (most often in the vertical position). You must experiment how much to tighten it.

For us, the rod itself cannot be removed from the rod receptacle. We think it may be super glued. This presents a problem because the equation to calculate gains needs the masses of the cart and rod individually. To avoid this problem, we founf all our gains through trial and error.

If using a sensitive camera such as our Day/Night camera, it is recommended to tape the bottom of the rod black to prevent reflections from the bright LED. You will be able to see on the Black & White image on the computer connected to the cameras if the program is picking up any extra light than the LED. There should only be a white circle indicating the LED position. Everything else must be covered or blocked, because these reflections throw off our tracking program, since it looks for the bright spot in the image.

4.5 Framegrabber Cards

S.Ha.R.K. 1.5 alpha supports the following framegrabber cards:

```
Hauppauge WinTV
Osprey-100, 200, 500, 2000, 540
ATI TV Wonder
Leadtek WinFast TV 2000, WinFast VC 100
STB TV PCI FM, Gateway P/N 6000704
I-O Data Co. GV-BCTV3/PCI, GV-BCTV4/PCI, GV-BCTV5/PCI
Pinnacle PCTV
3Dfx VoodooTV FM/ VoodooTV 200, VoodooTV 100/ STB OEM"
(Askey Magic/others) TView99 CPH06x,  TView99 CPH05x,  TView99 CPH061/06L
  (T1/LC), Askey CPH050
AVerMedia TVPhone98, TVCapture 98
VDOMATE TV TUNER CARD
Phoebe TV Master (CPH060)
Terratec TValue (Philips PAL B/G), Terratec TValue (Temic PAL B/G),
  Terratec Tvalue (Philips PAL I), Terratec TValue (Temic PAL I),
  Terratec TV Radio+, Terratec TV+ (V1.05), Terratec TValue (LR102),
```

```
    Terratec TValue Radio
  Zoltrix Genie TV
  Provideo PV143A, PV143B, PV143C, PV143D, PV150A-1, PV150A-2, PV150A-3,
    PV150A-4, PV150B-1, PV150B-2, PV150B-3, PV150B-4
  IVC-100, IVC-200, IVC-200G, IVC-120G
  GrandTec Multi Capture, Grand X-Guard
  Modular Technology MM201/MM202/MM205/MM210/MM215 PCTV
  FlyVideo 98 (LR50)/ Chronos Video Shuttle II, 98EZ (LR51)/ CyberMail
    AV, 98FM (LR50)/ Typhoon TView TV/FM Tuner
  Sensoray 311
  Canopus WinDVR PCI
  Face to Face Tvmax
  Nebula Electronics DigiTV
  Imagenation PXC200
```

A complete and up-to-date list can be found in folder [shark_root_directory]/drivers/bttv/bttv-cards.c

We used the Winfast TV2000 Deluxe (TV2000 with radio tuner) and the Osprey 101 (v. 100 card with a webcam). It is recommended to test yours first with the S.Ha.R.K. BTTV demo to make sure S.Ha.R.K. supports it. The only parameter you would have to know would be the channel which the card outputs the images from. Osprey 101 uses channel 0, while Winfast TV2000 uses channel 1. S.Ha.R.K. supports channels 0-4. If you don't know the channel your card uses, just try them sequentially. If it's the wrong channel, you will get either nothing or static.

Since most framegrabbers run at 30fps, you should schedule your framegrabber task accordingly.

4.6 Ethernet Cards

Not all Ethernet cards work with S.Ha.R.K. Test yours with the Talk demo found in the [shark_root_directory]/demos/network folder.

4.7 Analog Cameras

For our program, both the simple webcam and the super sensitive Day/Night security camera worked fine. As long it can connect to the framegrabber, your camera should work. NOTE: Your camera can be a webcam, security cam, camcorder, digital cam, etc.

4.8 Motes

MICA2 motes were used for this project. Motes are small wireless transmitters and receivers. These are made by Crossbow (http://www.xbow.com/Products/productsdetails.aspx?sid=72). They were programmed using TinyOS version 1.1.0 (http://www.tinyos.net). A schedule for

transmitting was determined beforehand and programmed into the motes. Each computer had a serial cable connected to COM1 and a mote programming board, which had a mote on it. Questions about the motes should be directed to Spencer Hoke (hoke@uiuc.edu).

Questions? Email: taoluo@engineering.uiuc.edu

5 S.Ha.R.K. DEVELOPMENT NOTES

Here you will find some notes explaining our code and some tips to help you avoid the problems we ran into while coding the S.Ha.R.K. portion of the project. It is recommended you read through everything sequentially. The main functions are discussed, so you can follow along through the code.

Our progression through S.Ha.R.K. versions:
Began with S.Ha.R.K. 1.3
S.Ha.R.K. 1.3 had conflicts with framegrabber cards and the DT2811 cards
Moved to S.Ha.R.K. 1.4
S.Ha.R.K. 1.4 had conflicts with framegrabber cards and the Ethernet cards
Moved to S.Ha.R.K. 1.5 Alpha
S.Ha.R.K. 1.5 Alpha had conflicts between Ethernet cards and USB ports
Moved to S.Ha.R.K. 1.5 Alpha + patch
Currently using S.Ha.R.K. 1.5 Alpha + patch

5.1 Important S.Ha.R.K. Types

5.1.1 Main Task

The *int main* function is a task in S.Ha.R.K., and it is scheduled as a Non-Real Time task. Thus, it has the lowest priority out of all the tasks. So if tasks don't get initialized or keyboard won't accept inputs (assuming the input code is part of int main), another task may be preventing main from running due to the low priority.

5.1.2 Hard Tasks

A hard task is given a period and maximum execution time. If a hard task does not finish before it's deadline, the program will halt and an error message will be printed. Hard tasks should only be used for events that must be completed by their deadline. If missing a deadline should not stop the program, use a soft task instead.

5.1.3 Soft Tasks

In S.Ha.R.K., soft task are not just hard task which ignores execution time overruns. Instead, soft tasks are scheduled using their own server (usually CBS). Also, they don't use deadlines like with hard tasks, they use mean execution times. Therefore, it is possible for soft tasks to consume all of the CPU resources, preventing lower tasks (like int main) from ever running.

5.1.4 Cabs

Cabs allow for one writer and multiple readers. We recommend using cabs instead of mutexes. Read our code or the official documentation to learn how to use cabs.

5.2 Adding Drivers (libdep.mk)

Any drivers you add in the drivers folder must be added into config/libdep.mk. The format is the following (adding myDriver):

```
# myDriver #
-----------------------------------------------------------------
ifeq ($(findstring __myDriver__,$(USELIB)) , __myDriver__)

INCL += -I$(BASE)/drivers/myDriver/include

ifeq ($(LIB_PATH)/libmyDriver.a,$(wildcard
$(LIB\_PATH)/libmyDriver.a)) LINK_LIB += -lmyDriver LIB_DEP +=
$(LIB\_PATH)/libmyDriver.a endif

endif
```

To use myDriver, add __myDriver__ to SHARKOPT of the makefile.
ie. SHARKOPT=¨__myDriver__¨

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.3 DT2811 A/D Card Driver (dt2811.c, dt2811.h)

IMPORTANT NOTE: The card only supports a range from -5 to 5 volts. Anything beyond these ranges will register as either the min or the max. The output from our potentiometer can exceed these ranges, thus we had to be careful in our experiments. We recommend you test first using the calibration phase of our programs (any except the first program) to find the min and the max.

S.Ha.R.K. did not supply a driver for this card, so we wrote our own. If you are not interested in writing a new driver, you can skip the rest of this section.

Read the AD1200 Reference Manual for an idea on how the DT2811 card works. We could not locate the DT2811 Manual, so we use the 100% compatible AD1200 Manual.

The main points of interest are the registers for reading and writing to the card. Finding the addresses of these registers should be of utmost importance. We believe that for ISA cards, the addresses are set on the card itself. Once the addresses are found, it is a simple task of writing and reading from these registers. For reading, first output the mode you want (we use mode 0) to the control register. Once the 7th bit in the control register is set (A/D is done), you can read from the low and high byte to get the data. For writing to the card, you just need to split your value into the low and high byte and output to the D/A register.

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.4 Stage 1: IPC_DT2811

This program is the basic inverted pendulum stabilizing controller. It reads from both built-in potentiometers to get the rod's angle and cart's x position, then calculates the voltage to output to move the cart. It communicates with only the DT2811 card. This demo is the base of all our other demos, so the code frequently reappears with minor adjustments.

5.4.1 void waitenter

This function appears in every later program. It stops the program from running until the Enter key is press. Usually, one Enter key press can finish two waitenter calls (sending CR+LF), so it might be necessary to call waitenter twice sequentially.

5.4.2 int da_motor

This function appears in every later program. It just takes in one float and outputs it to the correct channel of the dt2811 card.

5.4.3 void calibrate

This function appears in every later program. It is called once during the start of our program, and it stores the potentiometer inputs at several different points. This allows our conversions functions to operate. Later on, it will be modified to accept camera inputs and network inputs, but its job is still the same; storing initial data so that the conversions functions can operate.

5.4.4 float v2theta, v2x

These conversion functions appear in every later program. They stand for voltage-to-theta and voltage-to-x. They are called every time a task needs to convert raw voltages into degrees or cm. Later on, v2x gets renamed to input2x since the types of inputs increased from just voltages to pixels (from cameras), but its job remains the same.

5.4.5 TASK readingReg

This task is purely for debugging. It just displays the inputs and outputs.

5.4.6 TASK cart or cart_control

This task appears in every later program. This is the brain of our program. It reads the inputs and determines the output. We used both proportional and derivative control, but could not get integral control to work correctly. For Stage 5, the control functions were rewritten so that integral control worked properly and other improvements were made. The 6 gains used in the calculations (kp, kd, ki for angle and x) were found through trial and error. A better way would have been through calculations, but we could not remove the rod from the cart to get their separate masses.

Built into the controller is the limit on the output to between -5 and 5 volts, and there is also a safety clause, which stops the cart if the rod angles goes past 30 ° in either direction. Without this, the cart might get damaged by trying to move off the track. If the angle goes past 30 ° during normal control, then it

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.5 Framegrabber Code (camera.c, camera.h)

Most of the code was taken from the BTTV demo. The function start_frame_grabber starts the framegrabber, and grab_command retrieves the image from the hardware. The task draw_image_task reads from the framebuffer and draws the image pixel by pixel onto the screen. Our task, track_led_task, also reads from the framebuffer, but only draws the pixels that exceed the threshold. Next, it looks for the boundaries of the drawn pixels, and averages them to determine the LED position. Therefore, it is extremely important that there are no sources of light or reflections that exceed the threshold besides the LED itself.

When using the framegrabber, grx_text must be used to display text on the screen. No other print methods (cprintf, printf_xy) will work. In the makefile, SHARKOPT must have __FB__ __I2C__ __BTTV__. In the initfile, you must init FB26, I2C26, and BTTV26. Look in our code for the exact method.

The image that is outputted to the screen is 320 by 240, since most framegrabbers only output to that resolution. One thing though, most of the time it outputs only a partial image to the screen. 80% down the image there will be a line, and the rest of the picture below that will be the upper part of first image ever outputted to the screen (as in it never updates). We can't describe it very well, so your best bet would be to test out the S.Ha.R.K. BTTV demo (it happens in the official demo). By rebooting, there is a chance that it will disappear and we get a complete image. But it doesn't really bother our program as long as the bottom part is dark (so track_led_task doesn't pick it up). If it is not dark, try turning the camera on and off or the room lights on before turning the camera on.

Finally, all the framegrabber tasks should be soft, since drawing to the screen is time consuming and if it misses, we don't want the control to stop.

Questions? Email: jha4@engineering.uiuc.edu

5.6 Stage 2: IPC_Camera

We added a camera input to IPC_DT2811 to get IPC_Camera. Everything is pretty much the same, except it now uses a camera for the horizontal data instead of the potentiometer.

5.6.1 TASK display_inputs

This task appears in every later program. It runs only in the calibration phase for debugging purposes. You can look at the values printed to make sure everything is ok. Once calibration is done, a flag is set and the task exits out of the while loop. We did this because we could not get task_kill to work from another task.

Questions? Email: taoluo@engineering.uiuc.edu jha4@engineering.uiuc.edu

5.7 Stage 3: IPC_RC

IPC_DT2811 was split into two different programs, an I/O (IPC_IO) and a controller (IPC_Controller). They communicate using UDP on the Ethernet network. One computer running IPC_IO reads potentiometer inputs and sends them over the Ethernet to another computer running IPC_Controller. The controller calculated the proper output voltage and sends it back to the first computer, which outputs it to the cart.

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.8 Ethernet Network (network.c, network.h)

These functions are adapted from code found in the official Talk demo. The sender and receiver's IP addresses must be known for network to work. Both send and receive tasks should be soft.

5.8.1 FRAME *make_packet

This function constructs a packet to be sent over the Ethernet network. A packet is not necessary, but we felt it would be cleaner.

5.8.2 TASK network_send

Takes the packet from make_packet, and sends it over the Ethernet network.

5.8.3 TASK network_receive

Receives the packet from the Ethernet network and puts it into a cab.

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.9 Stage 4: IPC_NETCAM

We took IPC_Camera and made the camera sensors remote. The camera sensors (IPC_NETCamera) send their data through the Ethernet using UDP to the controller (IPC_NETController). The controller calculates the control output and sends it to the cart, which is connected to the controller computer. The network code is split into two files: network_receiver and

network_sender. IPC_NETCamera uses network_sender, while IPC_NETController uses network_receiver, because the sending is only done in one direction

Questions? Email: taoluo@engineering.uiuc.edu, jha4@engineering.uiuc.edu

5.10 IPC_MOTECAM

IPC_MOTECAM is the same as IPC_NETCAM except the Ethernet network code is replaced with mote network code.

Questions? Email: hoke@uiuc.edu

5.11 MOTES (motecom.c, motecom.h)

These files were supplied by Spencer Hoke, who programmed the wireless motes.

Questions? Email: hoke@uiuc.edu

6 TinyOS DEVELOPMENT NOTES

Here you will find some notes explaining our code and some tips to help you avoid the problems we ran into while coding the TinyOS portion of the project. We were using Mica2 motes for this project. It is recommended you read through everything sequentially.

6.1 TinyOS

"TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks." -http://www.tinyos.net

6.1.1 Where to Get Help

If you have never used TinyOS before, the tutorial available at http://www.tinyos.net/tinyos-1.x/doc/tutorial/index.html is highly recommended. Work through each section to go through examples of how to write programs and use the motes.

6.1.2 Using the Motes

To program the motes, attach them to a mote programming board. Then connect it to a computer that has TinyOS installed through the parallel port. Avoid touching the circuitry of the mote and programming boards as they can be easily damaged. Turn the mote on; a red LED should light up on the programming board. Go to the folder that contains the application you want to program (FAIEDF for instance) and type 'make mica2 install.[ID]' where ID will be the mote's ID.
Folder FAIEDF contains the implementation of the protocol. Use folder FAIEDFBase to create a receiver node (similar to TOSBase) which can monitor traffic forward it to a computer.

6.2 FAI-EDF Implementation

FAI-EDF stands for fault tolerant, asynchronous, implicit earliest deadline first. It is real-time wireless communication protocol developed by the Real-Time Systems group at the University of Illinois at Urbana-Champaign to guarantee deadlines even in the event of node failures and with no strict time synchronization between nodes.
This protocol has been implemented in the MAC layer of the motes.

6.2.1 Graphical Overview

To see a graphical overview of the system, run 'make mica2 docs' from the FAIEDF folder and then view the page
at: [tinyOS root]/tinyos-1.x/doc/nesdoc/mica2/apps.FAIEDF.FAIEDF.nc.app.html

6.2.2 Schedule and Message Formats

A schedule contains the order to transmit, with each entry representing one maximum-sized packet. If the entry is the mote's own ID, then the taskID contains the corresponding message to send.

```
Example Schedule at Mote A:
 +--------+---+---+---+---+---+---+---+
 | MoteID | A | A | B | B | A | C | A |
 +--------+---+---+---+---+---+---+---+
 | TaskID | 0 | 0 | x | x | 1 | x | 2 |
 +--------+---+---+---+---+---+---+---+
Example MsgTable at Mote A:
 +----------------+----------------+----------------+
 | Task 0 TOS_Msg | Task 1 TOS_Msg | Task 2 TOS_Msg |
 +----------------+----------------+----------------+
```

6.2.3 TOS_Msg Preparation

1) Fill in addr, type, and group fields
2) Fill in the data section with up to 24 bytes (Extra bytes will be filled with Local Address, PacketID, and extra budget)
3) Fill in the length field for the length of data you added (Setting length to 0 will send the minimum length packet, without corrupting the data, so budget can be forwarded)
4) Leave CRC alone - will be added automatically

6.2.4 Changing Messages

Be sure to stop the radio if changing the schedule. For updating message data, just update the table using 'atomic' around the value assignment. I use interface RTUpdate for this. Set the length to 0 to send a minimum length packet.

6.2.5 Maximum Sizes

The motes and this implementation allow for:
255 moteIDs (ID's 0-254)
256 tasks/mote (ID's 0-255)
4000 messages in schedule, with 4 tasks/mote (8K RAM)

6.2.6 LED Meanings

	On	Toggle
Red	Idle Channel Detected	n/a
Green	This Node is Doing Recovery	n/a
Yellow	n/a	Error Occurred

(But if they are never on, then they were disabled to save processing time)

6.2.7 Packet Format

```
Packet Format:
  Preamble      10 bytes
  Synch          2 bytes
  Header         6 bytes
    -Type          (1 byte)
    -State         (2 bytes)
    -Budget        (2 bytes)
    -Length        (1 byte)
  Payload        0-28 bytes variable
  CRC            2 bytes
  -------------------------
  Total Size    20 bytes + payload
```

6.2.8 Changing Settings

Set schedule and message table in FAIEDFM.nc
Set protocol parameters in FAIEDFSched.h
Additional settings are in /tos/platform/mica2/CC1000Const.h

6.2.9 Changing Preamble Size

I did reduce preamble size from 28 down to 10 in CC1000Const.h to speed up the protocol transmission. (¡10 did not work) This change also affects the standard protocol because it is not in the current folder, but only makes the standard one go faster.

6.2.10 When setting MAX_TX_TIME

-Use 25 + max payload as a minimum (extra 4 bytes for radio transition and processing + 1 for flush state if same mote xmits twice in a row)
-Aperiodic messages also require at least 25 bytes + payload, so if needed, the MAX_TX_TIME should be large

-Max packets/sec transmission is approximately 2400/MAX_TX_TIME
-Budget is passed to the next node and used up by sending a control message to hold the medium. Budget is also used up at the end of each hyperperiod if the same node starts and stops the schedule.

6.2.11 Real-Time Updating of Data from Serial Port

-This protocol is meant to be connected to a data source that sends update packets across the serial port.
-Using Interface RTUpdate, incoming packets are interpreted and the appropriate data is overwritten with the new data.
-Start and stop radio transmission functions are available (see FAIEDFSched.h for list of packet types)
-Incoming data must be spread out or bytes from the serial port will be overwritten. i.e. if two bytes arrive on serial port during the processing of a byte from the radio, the first byte will be dropped.

6.2.12 Not Implemented in this Version

-Stale recovery states
-Aperiodic sending (structure left for future addition of this)
-Sleeping when idle
-Dynamic number of budget/flags bytes (this one always sends 2)
-Support for messages longer than 1 packet (must be split into two). Use function sendDone in FAIEDFM to handle this if you want.

Questions? Email: hoke@uiuc.edu

7 RUNNING THE PROGRAMS

7.1 Stage 1: DT2811

First, the controlling computer should be connected to the cart through the A/D card. Go to folder [shark root]/demos/IPC_DT2811 and type 'make'. Copy the executable created onto the controlling PC and run it. Follow on-screen instructions.

7.2 Stage 2: Camera

In addition to the setup of the previous stage, attach one of the cameras to the controller computer. Go to folder [shark root]/demos/IPC_Camera and type 'make'. Copy the executable created onto the controlling computer and run it. Follow on-screen instructions.

7.3 Stage 3: Remote Control

Now attach two computers to the internet or a local network. One computer should be connected to the cart through the A/D card. Go to folder [shark root]/demos/IPC_RC and type 'make'. This time there will be two executables created. Run IPC_IO on the computer connected to the cart and run IPC_Control on the other computer. Run both and follow on-screen instructions. (Note: If you are not running this in our lab, you will probably need to change the IP addresses in the code before compiling)

7.4 Stage 4: Network Camera

This section requires a third computer, also attached to the internet or a local network. One computer should be connected to the cart through the A/D card, the other two should be connected to cameras. Go to folder [shark root]/demos/IPC_NETCAM and type 'make'. There will be two executables created. Run IPC_NETController on the computer connected to the cart and run IPC_NETCamera on the other computers. Run both and follow on-screen instructions.

7.5 Stage 5: Motes

Use the same setup as Stage 4, but this time do not connect the computers to the network. Plug a mote attached to a mote programming board into serial port COM1 on each computer. Program the motes as discussed in the previous section. The mote plugged into the controller should be programmed as a receiver and the other two should be transmitters. Avoid touching the circuitry of the mote and programming boards as they can be easily damaged. Go to folder [shark root]/demos/IPC_MOTECAM and type 'make'. There will be two executables created. Run IPC_MOTEController on the computer connected to the cart and run IPC_MOTECamera on the other computers. Run both and follow on-screen instructions for calibration.

Questions? Email: hoke@uiuc.edu

8 TROUBLESHOOTING

In this section you will find problems we went through and how we fixed them.

8.1 Problems and Solutions

8.1.1 Floppy disk died

SOLUTION
Don't run S.Ha.R.K. with floppy disks for an extended amount of time (i.e. overnight).
DETAILED SOLUTION
When running S.Ha.R.K. from a floppy disk, you will notice that the floppy disk access light is always on. S.Ha.R.K. never tells the disk to stop spinning. Ejecting the disk after the program starts works fine. We also recommend copying the files to the HD first if you will be running the program for a long time.

8.1.2 Rod is sticking when it's close to perpendicular

SOLUTION
Loosen rod receptacle screw.
DETAILED SOLUTION
Use 3/32 Allen wrench to loosen the screw a little bit. Only a tiny bit is needed.

8.1.3 Rod voltages changes after moving back to the same position

SOLUTION
Tighten rod receptacle screw.
DETAILED SOLUTION
Use 3/32 hex screwdriver to tighten the screw a little bit. Only a tiny bit is needed.

8.1.4 Rod potentiometer range is not as desired

SOLUTION
Adjust the potentiometer screw to change the voltage.
DETAILED SOLUTION
While holding the rod still, use a flathead screwdriver to adjust the potentiometer screw at the base of the rod receptacle.

8.1.5 Cart potentiometer range is not as desired

SOLUTION
Adjust the potentiometer screw to change the voltage.
DETAILED SOLUTION
While holding the rod still, use a flathead screwdriver to adjust the potentiometer screw holding the big wheel in place.

8.1.6 Cart moves even when the program is not running

SOLUTION
Send 0 volts to the cart.
DETAILED SOLUTION
Run any of our programs; the cart is stopped during the calibration phase. Or, in your code, have the card send out 0 volts (assuming range is 5 to -5) to stop the cart. You should always stop/send 0 volts to the cart before shutting down any program.

8.1.7 Cart balances, but then slowly moves off in one direction

SOLUTION
Track needs to be levelled.
DETAILED SOLUTION
Adjust the tracks height with the two screws hold the ends of the track (7/64 hex) or the four stands below the wooden base. If the track is level and the problem persists, play around with the track, perhaps unevening the track will fix this problem.

8.1.8 Cart and rod stop balancing with working program

SOLUTION
The track hardware changed and needs to be recalibrated.
DETAILED SOLUTION
Because of use, the hardware will change over time. Screws loosen, track became uneven, etc. You will just have to recheck each one by one. Since these changes also affect the control gains, you will most likely need to find any calibration values and the gains again.

8.1.9 Ethernet network does not work

SOLUTION
If Ethernet cable is securely in place, S.Ha.R.K. may not support your network card.
DETAILED SOLUTION
Test out your Ethernet card using official demos such as Talk or Network. Also, make sure you only have one active Ethernet card. You can disable Ethernet cards by either removing them for PCI cards or disabling them through the BIOS for cards built-in to the motherboard.

8.1.10 After exiting, the screen becomes gibberish

SOLUTION
Reboot.
DETAILED SOLUTION
This only happens when using BTTV in the demo and on only one of our computers (Dell). It doesn't really affect our testing, since we can hit the up arrow key in DOS to restore the last command. But rebooting will fix that problem.

8.1.11 Restarting S.Ha.R.K freezes computer during S.Ha.R.K. initialization

SOLUTION
Reboot.
DETAILED SOLUTION
This happens on one of our computers (Dell) but works fine on the other two. You will have to reboot the computer after exiting the demo.

8.1.12 The image from the framegrabber is not desired

SOLUTION
Reboot.
DETAILED SOLUTION
S.Ha.R.K. sometimes initializes the framegrabber incorrectly. Reboot the computer until image is corrected. You might have to reboot several times. If that has no effect, then there might be something wrong with the S.Ha.R.K. driver for that card. We suggest using a popular card, like the Winfast TV 2000. Also, make sure the camera is connected properly.

8.1.13 Program freezes after a few seconds. May or may not quit

SOLUTION
Memory leak most likely.
DETAILED SOLUTION
We had a lot of trouble with malloc and while loops, even when we freed the memory properly. Our suggestion is to avoid using malloc whenever possible. If you did use it, make sure the memory is freed. MAKE SURE!

8.1.14 Program freezes or quits with a non-scheduling error

SOLUTION
Reading past the end of an array most likely.
DETAILED SOLUTION
Make sure you never read past the end of an array in your code. This occurred several times in our testing when we did not have an array enough to hold text string we printed to the screen.

8.1.15 Mote appears to be on, but the computer isn't receiving anything

SOLUTION
Mote may not be connected properly.
DETAILED SOLUTION
Ensure that the mote is securly in the programming board and that the serial cable is not loose. The serial cable should be in port COM1 of the computer.

8.1.16 Program a mote gives a long list of flash errors

SOLUTION
Mote may not be connected properly.
DETAILED SOLUTION
Make sure that the batteries are good and that the mote has is connected securly in the programming board. The parallel cable is used for programming, so make sure it is not loose. If that does not work, try running MicaHWVerify on the mote. This will cause the LEDs to blink as a binary counter. If the LEDs don't work, then the mote is probably damaged.